# Chapter 1

# Introduction

Suppose we have a set of objects, for example the handwritten digits 0 and 1. Normally, they will be stored in the format of grey scale digitized images in a uniform size such as $28 \times 28$ and well aligned. To represent them, we will naturally convert them into vectors by concatenating the intensities of pixels leading to a 784 dimensional vector for an image. However, it is obvious that with such high dimensionality, we can barely understand the relational structure among them. It then gives rise to the searching for the representatives of these images in much lower dimensional space where we can interpret their relationships easily. This is what we call the problem of dimensionality reduction (DR). There are many definitions for DR from different point of view. We take the simplest one as

**Dimensionality Reduction:** Given a set of data $\{\mathbf{y}_i\}_{i=1}^N \in \mathbb{R}^D$, the problem of dimensionality reduction (DR) is to find their representation $\{\mathbf{x}_i\}_{i=1}^N \in \mathbb{R}^d$ with $d \ll D$.

The concept of relationship is inevitably involved which will basically be expressed by certain measure of either similarity or dissimilarity for instance distance metric. Intuitively, the Euclidean distance of the $i$th and $j$th images applies here to evaluate the dissimilarity denoted by $s_{ij}$. Assume we already have the representatives of these handwritten digits in low dimensional space, we would believe that the pairwise distances are reasonably preserved, i.e. $d_{ij} = s_{ij}$ where $d_{ij}$ is the distance in low dimensional space. However, owing to the loss of the degree of freedom from the greatly reduced dimensionality, we cannot expect the preservation to be exact. So we

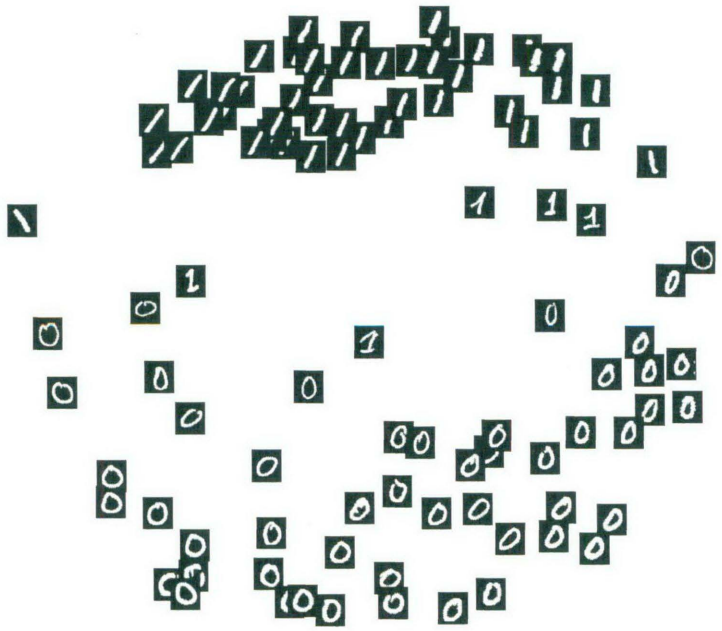

Figure 1.1: An illustration of dimensionality reduction. 100 1's and 0's are plotted here (50 each) to show the relational structure of these digits.

resort to the minimization of the inconsistency between these two distance measures according to which the optimal configuration is found by virtue of optimization techniques. This idea is concretised as the well known MultiDimensional Scaling (MDS) [25] algorithm, the first method addressing the dimensionality reduction problem in a principle manner. It was originated from mathematical psychology and extended into other areas rapidly afterwards. This simple idea of distance matching along with a workable algorithm enables us to find the representatives of handwritten digits in arbitrary dimensional space. We then plot them in a 2D plane as shown in Figure 1.1 where we can figure out the relations among those digits directly. In Figure 1.1, different 1's and 0's form two clear groups and similar pairs locate close to each other. This can also be observed from the pairwise distances $s_{ij}$ whereas it is possible only when the pool of the objects is rather small.

The similar scenarios appear frequently in machine learning area. Dimensionality reduction

has become one of important preprocessing steps for following tasks such as pattern recognition, classification, regression etc. The data range from surveillance video clips to web texts. A common feature of these data is that they all reside in very high dimensional spaces. The dimensionality grows exponentially with the complexity of the objects. For the handwritten digits in the former example, it is 784. As we shall see later, the vector representation for a piece of news could be more than 6000! Unfortunately, there is significant proportion of the dimensions are redundant in describing the nature of the objects. They will not only take too much storage space, but also pose serious challenge to processing algorithms caused by the "curse of dimensionality". For example, in polynomial regression of order $m$, the number of independent coefficients is $\binom{D+m}{m}$ which rapidly breaks down this method in the setting that the dimension $D$ is too large. Another extreme case is in the neural networks. The amount of the training data required increases exponentially with the number of weights in the networks which is determined by the $D$ of input data. It is impractical with these algorithms without dimensionality reduction for this kind of data.

A natural question would follow is how to reduce the dimensionality. Actually, the recent years have witnessed a great advance in this area. A number of methods [63, 87, 44, 93, 1, 65, 14, 133, 134, 13, 41, 92, 108, 129, 116, 31, 89, 16, 78, 137, 57, 19, 17, 128, 146, 145, 32] have been proposed other than the MDS used in the handwritten digit visualization example. Different points of views to the same problem generate a considerable variety of forms which not only enriched the DR algorithms family, but also provided a profound understanding of the nature of the DR itself. Among them, some are scaling the dimension by varying the objective functions where simple metrics are adopted for data. Nevertheless, some are seeking the accurate descriptions of the relationships among the objects by referring to the manifold hypothesis. There are both deterministic models which have simple elegant structure and probabilistic extensions paying attention to improving the robustness to the noise at the cost of simplicity. The history of the development of DR algorithms tells us that they are evolving from linear to nonlinear, global to local and even turning to the integration of these directions. However, no matter how different they seem to be, a fundamental principle plays an important role in DR, that is, some features existing in input objects should be retained in their low dimensional representatives in

such a manner that the distortion or information loss is minimized as much as possible. In our example, we maintained the pairwise distances and the distortion was due to the inconsistency between the dimensions of these two Euclidean distance measures which was minimized. Yet there are quite a few of DR methods consider the local relational structure of each object which is faithfully reconstructed in low dimensional space. Carefully analyzing what the existing DR approaches behave will provide us the hints for exploring new algorithms.

Another very important issue we need to mention is the type of the input objects in DR. There is a kind of so-called non-vectorial data (or structured data) emerged and quickly taken the principal position in machine learning area. They are often a highly structured combination of features, strings of symbols, sequences of segmentations, mixtures of different modalities such as images (expressed as graphs or sets of components), graphs, organic molecules, protein sequences, web pages and so on. They are often more natural representations for real-world data than vectors. Non-vectorial data normally have no vectorial forms and are not readily converted to vectors as well. To deal with the non-vectorial data, various methods have been presented recently. One of them is to embed the structured data into a metric space which falls in the scope of this thesis. We will apply DR to non-vectorial data to uncover their intrinsic structure in low dimensional space. But most existing DR methods require the vectorial input with the assumption that the objects are already in some vector spaces that is violated in non-vectorial setting. We have to develop new algorithms taking into consideration of this special data type. As pointed out before, we shall at first select an appropriate descriptor to characterize the relationships among the non-vectorial data (distances, kernels [120, 42] etc.), and then determine which feature(s) to be preserved and how to perform the reconstruction in low dimensional space endowed with a suitable measure. These issues will be unfolded in this thesis at length. We will try to go into very details of the DR and focus on the methods we developed for non-vectorial data.

The thesis will be organized as follows. We start with a literature review of several typical methods which are interesting and instructive in Chapter 2. The understanding of these methods naturally leads to a series of new algorithms. In the following chapter, we will discuss the descriptors of the relationships among non-vectorial that will be adopted in our DR algorithms and

present a kernel specially designed for images and shapes. It is followed by a chapter introducing a simple method called *K*ernel *L*aplacian *E*igenmaps (KLE) which projects the non-vectorial data to low dimensional space through changing the structure of original Laplacian Eigenmaps a little. In Chapter 5, we generalize the idea of KLE by employing another kernel in low dimensional space and we call this algorithm *T*win *K*ernel *E*mbedding (TKE). This algorithm is then applied to the fingerprint visualization and kernel learning in Chapter 6. In Chapter 7 and 8, we focus on the out-of-sample extension of the TKE which is followed by an interesting application on personalized fingerprint searching and classification. Finally we draw a conclusion and discuss the future work in this direction.

# Chapter 2

# Literature Review

## 2.1 Overview

Dimensionality reduction (DR) is an important step in many advanced applications such as exploratory data analysis and manifold learning, etc. It has been successfully applied in many areas including robotics [50], information retrieval [55], biometrics [113, 18, 97, 106], and bioinformatics [131, 68, 36, 105]. As defined in the last chapter, the target of DR is mainly to find the corresponding counterparts of input data in a much lower dimensional space (usually Euclidean) without incurring significant information loss. In this chapter, we will review several typical DR algorithms. With great development in computational ability, there are tens of such DR methods proposed each year recently and the arsenal of DR tools has been largely expanded. We will not review them exhaustively here, but have chosen to describe in more detail those that are closely related to our work. There are also a number of DR reviews available. In a recent report by L.J.P. van der Maaten et al. [136], the authors analyzed some linear techniques and nonlinear techniques that are representatives of existing methods by comparing their performances on both synthetic and real datasets. [153] compared the SOM (Self-Organizing Map) family methods with other popular ones. In [10, 119], the authors gave the comprehensive introduction to the spectral methods. Some old surveys of the DR could also be a good start such as [38, 103] where quit a few of classical approaches were discussed at great length.

To facilitate the further explanation, the following notations as in Table 2.1 will be adopted throughout the whole thesis. Note that if the input data have vector forms or $\mathcal{Y}$ is a vector space, we can have the dimensionality of the data as $D$ and $D \gg d$. However, for non-vectorial data, we do not actually have the dimensionality. In this case, finding the embeddings of the data is to embed the input data into a low-dimensional space where the following tasks can be performed.

Table 2.1: Notation conventions used in this thesis

| | |
|---|---|
| $\mathcal{Y}$ | input space (ambient space or abstract object space) |
| $\mathcal{X}$ | latent space (low-dimensional space, usually Euclidean) |
| $i$ | the index |
| $N$ | the number of data |
| $\{\mathbf{y}_i\}_{i=1}^N$ | given date set in $\mathcal{Y}$ |
| $\{\mathbf{x}_i\}_{i=1}^N$ | embeddings (embedded data) in $\mathcal{X}$ |
| $D$ | dimension of $\mathcal{Y}$ (if it is defined) |
| $d$ | dimension of $\mathcal{X}$ |
| $\mathbf{Y}$ | the set of input data in $\mathcal{Y}$ (if $\mathcal{Y}$ is a vector space, $\mathbf{Y}$ would be a matrix) |
| $\mathbf{X}$ | the set of embedded data in $\mathcal{X}$ (it is normally a matrix with data in rows) |
| $\mathbf{a}$ | vector (in column) |
| $\mathbf{A}$ or $\{a_{ij}\}$ | matrix |
| $[\mathbf{A}]_{ij}$ or $a_{ij}$ | the element at $i$th row and $j$th column in matrix $\mathbf{A}$ |
| tr$[\mathbf{A}]$ | the trace of matrix $\mathbf{A}$ |
| $\langle \mathbf{a}, \mathbf{b} \rangle$ | the inner product between $\mathbf{a}$ and $\mathbf{b}$ |
| $\|\mathbf{a}\|$ | $L_2$ norm of vector $\mathbf{a}$ ($\|\mathbf{a} - \mathbf{b}\|$ is the Euclidean distance between $\mathbf{a}$ and $\mathbf{b}$) |
| $d_{ij}$ or $d(i, j)$ | the distance between object $i$ and $j$ |
| $\mathcal{N}(\mu, \Sigma)$ | normal distribution with mean $\mu$ and covariance matrix $\Sigma$ |

## 2.2 Taxonomy of DR

There are many classification methods for the DR methods. According to whether a linear mapping between $\mathbf{Y}$ and $\mathbf{X}$ exists, they can be classified as linear methods and nonlinear methods (or NLDR for short). It seems that all the efforts of the research on DR have been concentrated on NLDR. So we have very limited number of linear methods since it is recognized that the linearity assumption does not hold in most situations. Because NLDR is not limited by such assumption, it gradually becomes the mainstream.

Another standard to classify the DR methods is to examine whether the method focuses on global structure or local neighborhoods. Some methods pay only attention to the overall structure expressed by the given data set and reproduce it in low-dimensional space. This kind of methods are called global methods. However, sometimes the neglected local structures veiled by the global structure that the global methods can never see contain more interesting information which may determine the nature of the data. In this case, local methods do better than the global ones. They favor the locality or neighborhood structure of each point in input space and reconstructed it in the latent space as similar as possible. But before we know the intrinsic structure of the data or the input space, we cannot objectively judge which one is better. Even so, locality preserving methods have gained more popularity especially in manifold learning.

Basically the DR procedure belongs to unsupervised learning. What DR faces is to uncover the coordinates of a given data set in low-dimensional space. The labels of the given data seem to be superfluous. But, it is also interesting to include the label information in DR. A typical application is the classification where DR is regarded as an interim step. Because the classification is to find an optimal boundary or decision function, DR can be seen as finding the optimal space where the input data are well separated. As the emergence of the semi-supervised learning, DR can also be adapted to this setting by consider only part of the label information. It does make sense in the situation that the labels are extremely hard or expensive to get. These known labels can be incorporated into the DR procedure to provide valuable hints on how the embeddings should locate or what the latent space would be.

So far we only focused on a fixed data set. Nevertheless, it is necessary to consider the extension to the novel samples in real applications, for example the classification. In DR, this is called Out-Of-Sample (OOS) extension. It is also a problem attracting more and more attention in DR research. We need not only the embeddings for the given data, but also a mechanism for new inputs. For those methods without OOS extension, the only possibility to get the representatives of new arrivals is to start the algorithms all over again using the expanded data set which is time consuming and wasteful. The results found in the previous step will entirely be abandoned. More seriously, as the number of the inputs grows, the complexity would increase rapidly. This will be problematic in real time systems such as objects motion tracking where new data flood

in continuously. Unfortunately, not all DR methods have such ability to handle new input data even though some of them have satisfactory results for given data set. Consequently, there are researches on how to extend the existing methods to novel samples in different ways. Some DR methods have flexible structures such as neighborhood graph which can be exploited to accommodate new samples without reconstruction of the whole graph. Nonetheless, situations for others could not be easy to deal with. This issue will be discussed in Chapter 7 in more detail.

In addition to these classification standards for DR algorithms, some other standards still exist. For example, according to the underlying models from the statistical point of view, they can be classified to deterministic model or probabilistic model. Or to see whether there is a graph underpinning the method induce to graph based model and non-graph based model and so on. Though there are a variety of standards available, sometimes the boundary is ambiguous. Therefore, it is not necessary to group those methods so rigidly. It is just a tool facilitating us to understand some basic properties of these methods.

In following sections, we will discuss several popular and instructive DR algorithms. Through the analysis of these methods, we will comprehend the core part of the dimensionality reduction and its underlying principle.

## 2.3 Principal Component Analysis

Principal Component Analysis (PCA) [63] is perhaps the most widely used linear dimensionality reduction algorithm due to its simplicity. The basic assumption is that the input data can be projected onto a subspace spanned by $d$ orthonormal axes (the so-called principal components) where the covariance of the data is mostly preserved. This idea is achieved by minimizing the following reconstruction error

$$\sum_{i=1}^{N} \|\mathbf{y}_i - \widehat{\mathbf{y}}_i\|^2, \tag{2.1}$$

where $\widehat{\mathbf{y}}_i$ is the projection of $\mathbf{y}_i$ on the optimal orthonormal axes preserving the maximum variance in the data. That is, $\widehat{\mathbf{y}}_i = \overline{\mathbf{y}} + \sum_{j=1}^{d} \langle \mathbf{y}_i, \mathbf{e}_j \rangle \mathbf{e}_j$ where $\overline{\mathbf{y}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{y}_i$ is the mean of data $\mathbf{y}_i$ and $\mathbf{e}_j$ is the $j$-th principal component and $d$ the number of principal components. A constraint

from the orthonormality is $\langle \mathbf{e}_i, \mathbf{e}_j \rangle = \delta_{ij}$ where $\delta_{ij}$ is the Kronecker delta (i.e. $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ otherwise). Minimizing (2.1) is equivalent to maximizing $\sum_{i=1}^{N} \widehat{\mathbf{y}}_i^\top \widehat{\mathbf{y}}_i$ because of the distance preserving property of orthogonal transformations. It has been proved in [63] that the optimal orthonormal axes (principal components) can be obtained by solving a characteristic equation of an eigensystem of the form

$$\lambda \boldsymbol{v} = \mathbf{C}\boldsymbol{v}, \tag{2.2}$$

where $\mathbf{C}$ is the covariance matrix of the centered observations $\tilde{\mathbf{y}}_i$:

$$\mathbf{C} = \frac{1}{N} \sum_{i=1}^{N} \tilde{\mathbf{y}}_i \tilde{\mathbf{y}}_i^\top \tag{2.3}$$

where $\tilde{\mathbf{y}}_i = \mathbf{y}_i - \overline{\mathbf{y}}$. Denote $\mathbf{M} = [\boldsymbol{v}_1 \dots \boldsymbol{v}_d]$ the $D \times d$ matrix composed by those orthonormal leading eigenvectors of (2.2) corresponding to $d$ largest eigenvalues. $\mathbf{M}$ forms the linear mapping from input space to latent space, i.e.

$$\mathbf{x}_i = \mathbf{M}^\top \tilde{\mathbf{y}}_i.$$

To see the principal components retain most of the variance present in input data, consider the variance of the projections onto the first principal component:

$$\text{var}[\boldsymbol{v}_1^\top \tilde{\mathbf{y}}] = \boldsymbol{v}_1^\top \mathbf{C} \boldsymbol{v}_1 \tag{2.4}$$

which should be maximized subject to the condition $\boldsymbol{v}_1^\top \boldsymbol{v}_1 = 1$. Use the technique of Lagrange multipliers and maximize

$$\boldsymbol{v}_1^\top \mathbf{C} \boldsymbol{v}_1 - \lambda(\boldsymbol{v}_1^\top \boldsymbol{v}_1 - 1),$$

where $\lambda$ is a Lagrange multiplier. Differentiation with respect to $\boldsymbol{v}_1$ gives

$$\mathbf{C}\boldsymbol{v}_1 - \lambda \boldsymbol{v}_1 = 0,$$

or

$$\mathbf{C}\boldsymbol{v}_1 = \lambda \boldsymbol{v}_1. \tag{2.5}$$

Thus $\lambda$ is an eigenvalue of $\mathbf{C}$ and $\boldsymbol{v}_1$ is the corresponding eigenvector. Substitute (2.5) into (2.4), the variance is

$$\boldsymbol{v}_1^\top \mathbf{C} \boldsymbol{v}_1 = \boldsymbol{v}_1^\top \lambda \boldsymbol{v}_1 = \lambda \boldsymbol{v}_1^\top \boldsymbol{v}_1 = \lambda$$

so $\lambda$ must be as large as possible to maximize the variance. Therefore, $\lambda$ is the largest eigenvalue and hence $\boldsymbol{v}_1$ is the corresponding first principal component. For the following principal component, we have similar results.

If we substitute (2.3) into (2.2)

$$\lambda \boldsymbol{v} = \frac{1}{N} \sum_{i=1}^{N} \tilde{\mathbf{y}}_i \tilde{\mathbf{y}}_i^\top \boldsymbol{v},$$

it is clear that the eigenvector $\boldsymbol{v}$ is actually spanned by input data. So we can define

$$\boldsymbol{v} = \sum_{i=1}^{N} a_i \tilde{\mathbf{y}}_i = \widetilde{\mathbf{Y}}^\top \boldsymbol{\alpha}$$

where $\boldsymbol{\alpha} = [a_1, \ldots, a_N]^\top$ and $\widetilde{\mathbf{Y}} = [\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_2, ..., \tilde{\mathbf{y}}_N]^\top$, then (2.2) can be rewritten as the following eigensystem for $\boldsymbol{\alpha}$

$$N\lambda \boldsymbol{\alpha} = \widetilde{\mathbf{Y}} \widetilde{\mathbf{Y}}^\top \boldsymbol{\alpha} \tag{2.6}$$

Because $\langle \boldsymbol{v}, \boldsymbol{v} \rangle = 1$, we have

$$(\widetilde{\mathbf{Y}}^\top \boldsymbol{\alpha})^\top \widetilde{\mathbf{Y}}^\top \boldsymbol{\alpha} = \boldsymbol{\alpha}^\top \widetilde{\mathbf{Y}} \widetilde{\mathbf{Y}}^\top \boldsymbol{\alpha} = N\lambda \boldsymbol{\alpha}^\top \boldsymbol{\alpha} = 1$$

which leads to the normalization of the $\boldsymbol{\alpha}$ being $\boldsymbol{\alpha}/\sqrt{N\lambda}$. Let $\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$, ..., $\boldsymbol{\alpha}_d$ be the first $d$ leading eigenvectors of (2.6) corresponding to the principal components and eigenvalues. Denote $\underline{\boldsymbol{\alpha}} = [\sqrt{N\lambda_1}\boldsymbol{\alpha}_1, \sqrt{N\lambda_2}\boldsymbol{\alpha}_2, \ldots, \sqrt{N\lambda_d}\boldsymbol{\alpha}_d]$ which is an $N \times d$ matrix. It is actually the new coordinates (i.e. $\mathbf{X}$) in low-dimensional space spanned by principal components. To clarify this, the projections of $\mathbf{y}_i$'s are

$$\widetilde{\mathbf{Y}}\mathbf{M} = \widetilde{\mathbf{Y}} \widetilde{\mathbf{Y}}^\top [\frac{\boldsymbol{\alpha}_1}{\sqrt{N\lambda_1}}, \ldots, \frac{\boldsymbol{\alpha}_d}{\sqrt{N\lambda_d}}] = [\sqrt{N\lambda_1}\boldsymbol{\alpha}_1, \ldots, \sqrt{N\lambda_d}\boldsymbol{\alpha}_d]. \tag{2.7}$$

We will see that in next section, it is actually the result of classical scaling.

Since PCA assumes the linear relation in the input data, it may not work well when this condition is violated. This gives rise to the nonlinear version of PCA which is called Kernel PCA [121]. The idea is to map the input data into a feature space by a nonlinear mapping function where the PCA is carried out by resorting to the "kernel trick". We will take a look at it in section 2.11. Another observation is that the PCA takes into account all the input data at a time, so it is

a global method. Variants like local PCA proposed by Nandakishore Kambhatla et. al. [65] and topological local PCA by Zhiyong Liu et. al. [89] are capable of exploiting both global and local structure of the input data. Interestingly, the probabilistic view of the PCA leads to probabilistic PCA [134] and the corresponding probabilistic version of local PCA is mixtures of probabilistic PCA [133]. It is clear that different treatment will result in many variants of the PCA like sparse PCA [157], binary PCA [127] and so on. The research on PCA has been carried on for years and will continue in future. Its simple formulation and clear interpretation are always the inspiration for new methods.

## 2.4 Multi-Dimensional Scaling

Multi-Dimensional Scaling (MDS) [25] is a collection of methods preserving pairwise similarities. We shall start with the simplest classical scaling and then discuss the variants of MDS.

Classical scaling originated in the 1930s when it was shown that the coordinates of a set of points in an Euclidean space can be found that recovers the distances matrix of the given data set. Let $d_{ij}$ be the distance between $\mathbf{y}_i$ and $\mathbf{y}_j$, then

$$\begin{aligned} d_{ij}^2 &= \|\mathbf{y}_i - \mathbf{y}_j\|^2 = (\mathbf{y}_i - \mathbf{y}_j)^\top (\mathbf{y}_i - \mathbf{y}_j) \\ &= \mathbf{y}_i^\top \mathbf{y}_i + \mathbf{y}_j^\top \mathbf{y}_j - 2\mathbf{y}_i^\top \mathbf{y}_j. \end{aligned}$$

Let the inner product matrix be $\mathbf{B}$, where $[B]_{ij} = b_{ij} = \mathbf{y}_i^\top \mathbf{y}_j$. We have

$$\begin{aligned} b_{ij} &= \mathbf{y}_i^\top \mathbf{y}_j, \\ &= -\frac{1}{2}(d_{ij}^2 - \frac{1}{N}\sum_{i=1}^{N} d_{ij}^2 - \frac{1}{N}\sum_{j=1}^{N} d_{ij}^2 + \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}^2). \end{aligned}$$

Let $a_{ij} = -\frac{1}{2}d_{ij}^2$ and the matrix of $a_{ij}$ be $\mathbf{A}$ as $[\mathbf{A}]_{ij} = a_{ij}$, and hence the inner product matrix $\mathbf{B}$ is

$$\mathbf{B} = \mathbf{HAH} \tag{2.8}$$

where $\mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\top$ with $\mathbf{I}$ the identity matrix and $\mathbf{1}$ the column vector of $N$ ones. $\mathbf{H}$ is called the centering matrix and above operation is called doubly centering. Now the inner product

matrix $\mathbf{B}$ can be expressed as

$$\mathbf{B} = \mathbf{X}\mathbf{X}^\top,$$

which means the coordinates in $\mathbf{X}$ can recover the inner product matrix $\mathbf{B}$ and hence retain the pairwise distance between input data. Now $\mathbf{B}$ is written in terms of its spectral decomposition

$$\mathbf{B} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top, \tag{2.9}$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$, the diagonal matrix of ordered eigenvalues (in descending order) $\{\lambda_i\}$ of $\mathbf{B}$, and $\mathbf{V}$ the unitary matrix consisting of orthonormal corresponding eigenvectors and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N]$. Let $\mathbf{X} = \mathbf{V}\mathbf{\Lambda}^{\frac{1}{2}}$, then $\mathbf{X}$ can perfectly recover $\mathbf{B}$. In practice, we just select $d$ leading eigenvalues, that is

$$\mathbf{\Lambda}_1 = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d),\ \mathbf{V}_1 = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d],$$

and thus

$$\mathbf{X} = \mathbf{V}_1\mathbf{\Lambda}_1^{\frac{1}{2}}. \tag{2.10}$$

Then the difference brought by this choice is measured by

$$\sum_{ij}(b_{ij} - b^*_{ij})^2 = \text{tr}[\mathbf{B} - \mathbf{B}^*]^2 \tag{2.11}$$

where $\mathbf{B}^* = \mathbf{V}_1\mathbf{\Lambda}_1\mathbf{V}_1^\top$ the inner product matrix of $\mathbf{X}$. The above error will be

$$\sum_{i=d+1}^{N} \lambda_i^2. \tag{2.12}$$

Since $\mathbf{B}$ is positive semi-definite matrix, $\lambda_i \geq 0$. Selecting first $d$ leading eigenvectors will give us the minimum error. The advantage of using distance matrix is to embrace other dissimilarity measures. To be of practical use, a configuration of points needs to be found for a set of dissimilarities $\{\delta_{ij}\}$ rather than simply for the Euclidean distances between input data $\{d_{ij}\}$. It has been proved that when the doubly centered inner product matrix $\mathbf{B}$ derived from dissimilarity matrix $\mathbf{A}$ where $[\mathbf{A}]_{ij} = \delta_{ij}$ is positive semi-definite of rank $p$, then a configuration in $p$ dimensional Euclidean space can be found that preserves the dissimilarity.

Combining equations (2.9), (2.10), we can see that $\mathbf{V}$ is the principal components of $\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^\top$ and thus

$$\begin{aligned}\mathbf{X} &= [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d]\text{diag}(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_d}) \\ &= [\sqrt{\lambda_1}\mathbf{v}_1, \sqrt{\lambda_2}\mathbf{v}_2, \ldots, \sqrt{\lambda_d}\mathbf{v}_d,],\end{aligned}$$

which is the same as the result of PCA as shown in (2.7) up to a scale $\sqrt{N}$. Because only inner products are involved in the formulae, we can also take this advantage together with the "kernel trick" to make MDS and PCA non-vectorial data applicable. This will be discussed later in section 2.11.

From (2.10), (2.11) and (2.12), we can see that the classical scaling is actually minimizing the following objective function

$$\sum_{ij}(\mathbf{y}_i^\top \mathbf{y}_j - \mathbf{x}_i^\top \mathbf{x}_j)^2,$$

where the $\mathbf{y}_i$'s have been centered. It can also be interpreted as to minimize the difference of two inner products: one is in input space and the other is in latent space. This least squares can be naturally extended to other measures such as dissimilarity such as Euclidean distances as

$$\sum_{ij}(\|\mathbf{y}_i - \mathbf{y}_j\| - \|\mathbf{x}_i - \mathbf{x}_j\|)^2. \tag{2.13}$$

Equation (2.13) is also called the stress function. Some variants of the stress functions has been proposed later on. One of them is the Sammon cost function [25] which emphasizes the locality to some extent

$$\sum_{i<j}\|\mathbf{y}_i - \mathbf{y}_j\|^{-1}(\|\mathbf{y}_i - \mathbf{y}_j\| - \|\mathbf{x}_i - \mathbf{x}_j\|)^2 / \sum_{i<j}\|\mathbf{y}_i - \mathbf{y}_j\|. \tag{2.14}$$

The weights introduced by the inverse of the distance put on the difference force the algorithm to pay more attention to those pairs inputs that are close to each other. The denominator, $\sum_{i<j}\|\mathbf{y}_i - \mathbf{y}_j\|$ is a normalizing term making the above cost function scale free.

Different objective functions require different optimization methods. For (2.14) iterative minimization such as gradient based method has to be employed while the simple eigendecomposition of a pairwise dissimilarity matrix could be applied to classical scaling. The MDS mentioned

above are the so-called metric MDS. There is also another category MDS called nonmetric MDS among which the Kruskal approach [73, 74] can be regarded as the foundation where a loss function was defined and numerical solution was provided.

## 2.5 Laplacian Eigenmaps

Laplacian Eigenmaps (LE) [6] is a typical nonlinear method that belongs to the family of graph-based DR methods. It attempts to preserve proximity relations in the input data which is expressed by a weight matrix based on adjacency graph (or called neighborhood graph). This adjacency graph $G$ is constructed by referring to $\varepsilon$ neighborhood or $n$ nearest neighbor criterion. An edge will connect $\mathbf{y}_j$ and $\mathbf{y}_i$ if

- $\|\mathbf{y}_i - \mathbf{y}_j\|^2 < \varepsilon$ ($\varepsilon$ neighborhood),
- or if $\mathbf{y}_j$ is among $n$ nearest neighbors of $\mathbf{y}_i$ and vice versa ($n$ nearest neighbor).

The advantages of $\varepsilon$ neighborhood include its geometric motivation and inherent symmetry. But it often leads to graphs with several connected components and the value of $\varepsilon$ is hard to determine in practice. A heuristic search has to be involved in choosing a suitable $\varepsilon$. Conversely, in $n$ nearest neighbor, the $n$ is easy to choose and it does not tend to end up with disconnected graphs. However, it will have less geometrical meaning than $\varepsilon$ neighborhood. Some methods such as [150] are proposed to construct a valid adjacency graph more efficiently. It is worth noting that the adjacency graph (or neighborhood graph) plays an important role in dimensionality reduction which leads to a series of graph based methods [119, 147].

After the construction of the adjacency graph, the weights on the edges are evaluated that reveals the proximity relations among the input data. There are also two variations of finding the weights in LE:

- exponential decay function:

$$\text{the weight } w_{ij} = \begin{cases} e^{-\sigma\|\mathbf{y}_i - \mathbf{y}_j\|^2}, \text{ if } \mathbf{y}_i \text{ is connected with } \mathbf{y}_j; \\ 0, \text{otherwise.} \end{cases}$$

- binary ($\sigma = 0$): $w_{ij} = 1$ if $\mathbf{y}_i$ and $\mathbf{y}_j$ are connected, and $w_{ij} = 0$ otherwise. This simplification avoids the need to choose $\sigma$.

The choice of the weights not only reflects the proximity among the data but also meets the requirement of the weights in spectral graph theory [22] which underpins this method, that is, the weights should be positive. The weight matrix $\mathbf{W}$ ($\{w_{ij}\}$) containing the proximity information is then included in an objective function to be minimized

$$\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\|\mathbf{x}_i - \mathbf{x}_j\|^2, \tag{2.15}$$

subject to $\mathbf{X}^\top\mathbf{D}\mathbf{X} = \mathbf{I}$ and $\mathbf{X}^\top\mathbf{D}\mathbf{1} = \mathbf{0}$ where the diagonal matrix $\mathbf{D}$ is derived from $\mathbf{W}$ with $[\mathbf{D}]_{ii} = \sum_{j=1}^{N} w_{ij}$. $\mathbf{1}$ and $\mathbf{0}$ are all 1 and 0 column vectors respectively. The constraint $\mathbf{X}^\top\mathbf{D}\mathbf{X} = \mathbf{1}$ removes an arbitrary scaling factor in the embeddings and $\mathbf{X}^\top\mathbf{D}\mathbf{1} = \mathbf{0}$ can be interpreted as removing a translation invariance in $\mathbf{X}$. The objective function with the choice of weights $w_{ij}$ incurs a heavy penalty if neighboring points $\mathbf{y}_i$ and $\mathbf{y}_j$ are mapped far apart. Therefore, minimizing it is an attempt to ensure that if $\mathbf{y}_i$ and $\mathbf{y}_j$ are "close", then $\mathbf{x}_i$ and $\mathbf{x}_j$ are close as well. It is obvious from equation (2.15) that Laplacian Eigenmaps matches the proximity structure of the input data to the Euclidean distances of their embeddings in the latent space. The embeddings are obtained by solving the generalized eigenvector problem

$$\mathbf{L}\boldsymbol{\upsilon} = \lambda\mathbf{D}\boldsymbol{\upsilon},$$

where $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is the Laplacian matrix which is a symmetric, positive semidefinite matrix. Suppose the eigenvalues are ordered in ascending order $0 = \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_{N-1}$ corresponding to eigenvectors $\boldsymbol{\upsilon}_0, \boldsymbol{\upsilon}_1, \ldots, \boldsymbol{\upsilon}_{N-1}$, the result is

$$\mathbf{X} = [\boldsymbol{\upsilon}_1, \ldots, \boldsymbol{\upsilon}_{d+1}]$$

left out the eigenvector $\upsilon_0$ corresponding to the eigenvalue 0.

LE is a local method since it is constructed on the adjacency graph. Several variants are derived from original LE such as the LPP (Locality Preserving Projection) [56] and the KLE (Kernel LE) [46]. LPP introduces a linear constraint between input data and embeddings, i.e

$\mathbf{x}_i = \mathbf{A}\mathbf{y}_i$ while KLE replaces the weight matrix by a sparse kernel Gram matrix. They are variants of the LE algorithm but attempting to solve different problems, that is LPP provides a solution for the out-of-sample problem (estimation of the embeddings for new input data [9]) and KLE is applicable to non-vectorial data by virtue of the kernel mapping. These two methods will be analyzed in detail in Chapter 4 and 7.

## 2.6 Isomap

The Isomap (Isometric feature mapping) proposed in [129] is a global method actually built on the classical MDS. It uses the geodesic distances rather than the Euclidean distances between all pairs of data points to capture the intrinsic geometry of the data. The algorithm consists of 3 steps.

The first step constructs a neighborhood graph with edges of weight $d_Y(i,j)$ (the Euclidean distance of $\mathbf{y}_i$ and $\mathbf{y}_j$) between the neighbouring points. The graph $G$ over all data points connects points $\mathbf{y}_i$ and $\mathbf{y}_j$ if (as measured by $d_Y(i,j)$) they are closer than $\varepsilon$ ($\varepsilon$-Isomap), or if $\mathbf{y}_i$ is one of the $n$ nearest neighbors of $\mathbf{y}_j$ ($n$-Isomap). To eliminate the guess work for $n$ or $\varepsilon$, Nathan and John [98] proposed an adaptive technique to find the optimal neighborhoods automatically. In the second step, Isomap estimates the geodesic distances $d_G(i,j)$ between all pairs of points by computing their shortest path distances in the graph. Finally $\mathbf{D}_G = \{d_G(i,j)\}$ is input to the classical scaling (see section 2.4) to find the embeddings. Hence the objective function to be minimized is

$$\|\tau(\mathbf{D}_G) - \tau(\mathbf{D}_X)\|_F^2 \tag{2.16}$$

where $\mathbf{D}_X = \{\|\mathbf{x}_i - \mathbf{x}_j\|\}$ is the Euclidean distance matrix. In order to comply with the conditions of MDS, the $\tau$ operator in (2.16) converts the distance to an inner product defined as $\tau(\mathbf{D}) = -\mathbf{HSH}/2$, where $\mathbf{S}$ is the matrix of squared distances $\{s_{ij} = d_{ij}^2\}$, and $\mathbf{H} = \mathbf{I} - \mathbf{1}/N$ the "centering matrix" exactly the same as the one in (2.8) ($\mathbf{I}$ is the identity matrix and $\mathbf{1}$ is the matrix with all 1 entries). In essence, Isomap is preserving the geodesic distances since equation (2.16) attempts to constraint $\|\mathbf{x}_i - \mathbf{x}_j\| = d_G(i,j)$.

Isomap is guaranteed to recover the true dimensionality and geometric structure of a strictly larger class of nonlinear manifolds asymptotically. This guarantee of asymptotic convergence rests on a proof that as the number of data points increases, the graph distances $d_G(i,j)$ provide increasingly better approximations to the intrinsic geodesic distances $d_M(i,j)$, becoming arbitrarily accurate in the limit of infinite data. However, the sample size required to estimate geodesic distance accurately may be impractically large as the manifold where the data points lie on or near to is very irregular exhibiting extreme values or deviating from a uniform density.

By improving several aspects of the Isomap, many variants emerged recently. Having observed that certain standard phenomena in images such as occlusion, non-convexity of the underlying parametrization can prevent Isomap from working perfectly [35], David and Carrie proposed the local Isomap [34] that uses the same geodesic distances as Isomap but only in a local fashion. Also, after comparing advantages and disadvantages of global and local methods, Vin de and Joshua presented Conformal Isomap and Landmark Isomap [30] as a combination of global and local methods with the virtues of computational sparsity and ability to invert conformal maps. Moreover, [11, 154, 61] extended the Isomap to novel input samples. As the topological instability of Isomap was revealed in [5], robust kernel Isomap was then proposed [21]. Furthermore, its drawbacks of topological instability and overly strong assumptions on convexity and global isometry also leads to new algorithms, say Hessian LLE [33] (see section 2.8).

## 2.7 Locally Linear Embedding

Proposed in [116], Locally Linear Embedding (LLE) preserves the local linear relations in the input data. This is accomplished by capturing the proximity information in a weight matrix $\mathbf{W}$ that is used in minimizing the reconstruction error defined by

$$\sum_i \|\mathbf{x}_i - \sum_j w_{ij}\mathbf{x}_j\|^2, \tag{2.17}$$

where $[\mathbf{W}]_{ij} = w_{ij}$ is the weight derived from a neighborhood graph. There are also 3 steps in LLE: First, assign neighbors to each data point $\mathbf{y}_i$ (for example by using the $n$ nearest neighbors). Second, compute the weights $w_{ij}$ that best linearly reconstruct $\mathbf{y}_i$ from its neighbors by minimizing the following constrained least-squares problem

$$\mathbf{W} = \arg\min_{\mathbf{W}} \sum_{i=1}^{N} \|\mathbf{y}_i - \sum_{j=1}^{N} w_{ij}\mathbf{y}_j\|^2$$

subject to $\sum_j w_{ij} = 1$ and $w_{ij} = 0$ if there is no edge between $\mathbf{y}_i$ and $\mathbf{y}_i$ in the neighborhood graph. In essence, the weight matrix $\mathbf{W}$ depicts the similarity relations in the input data that is to be enforced in the latent space. Finally compute the low-dimensional embedding vectors $\mathbf{X}$ best reconstructed by $w_{ij}$ by minimizing (2.17) under the constraints $\sum_i \mathbf{x}_i = \mathbf{0}$ and $\frac{1}{N}\sum_i \mathbf{x}_i\mathbf{x}_i^\top = \mathbf{I}$ to remove arbitrary translations of the embeddings and avoid degenerate solutions. The objective function can be written in a quadratic form

$$\sum_{ij} m_{ij}(\mathbf{x}_i^\top \mathbf{x}_j)$$

with $m_{ij} = \delta_{ij} - w_{ij} - w_{ji} + \sum_k w_{ki}w_{kj}$ and $\delta_{ij}$ is 1 if $i = j$ and 0 otherwise. The optimal embedding $\mathbf{X}$, up to a global rotation of the latent space, is found by computing the bottom $d+1$ eigenvectors (those corresponding to its smallest $d+1$ eigenvalues) of the matrix $\mathbf{M}$ ($\{m_{ij}\}$). The bottom eigenvector of this matrix, which we discard, is the unit vector with all equal components; it represents a free translation mode of eigenvalue zero. (Discarding it enforces the constraint that the embeddings have zero mean.)

Because the matrix $\mathbf{M}$ can be stored and manipulated as a sparse matrix, it brings substantial computational savings for large values of $N$. Moreover, its bottom $d+1$ eigenvectors can be found efficiently without performing a full matrix diagonalization. Hence LLE has low computational cost which is desirable in real applications. This feature was further extended in incremental locally linear embedding [72] where a much smaller optimization problem replaced the "batch" optimization. Although only linear algebra is employed in the algorithm, the constraint that points are only reconstructed from neighbors can result in highly nonlinear embeddings. Therefore, LLE is actually a nonlinear DR method.

Several problems exist in LLE. For example it is shown that it is sensitive to the outliers since in the presence of the noisy data which are not on the manifold, the $n$ nearest neighbor would no longer give an appropriate approximation of the linear patch. Hence robust LLE [19] was proposed. Another problem is non-convexity which occasionally occurs in computer vision for instance the occlusion, restricted areas etc. Furthermore, the assumption of the sufficiency of data is not clear for image manifolds.

## 2.8 Hessian Locally Linear Embedding

Hessian LLE (HLLE) [33] is closely related to LE in that HLLE replaces the graph Laplacian with a quadratic form based on the Hessian. HLLE addresses the problem inherent in Isomap such as the assumption of global isometry and convexity etc. HLLE derives from a conceptual framework of local isometry which is able to handle a significantly wider class of situations than the original Isomap algorithm. The solution of HLLE is inspired by LLE. They share quite similar procedures. HLLE also considers the linear patches of the manifold and maps them into a low dimensional space. However, in HLLE, this is done by warping the patches as little as possible. Practically, the "warping" is measured by the Frobenius norm of the Hessian matrix $\mathcal{H}(f)$ which is minimized through solving an eigenvalue problem.

HLLE algorithm has 3 steps. First, find the nearest neighbors which is identical to LLE. Second, for each point on the manifold, perform the PCA of neighboring points to estimate the tangent space and then develop the least-squares estimation of the Hessian. Finally, optimal solution takes rows of $\mathbf{X}$ as the eigenvectors of $\mathcal{H}(f)$ with minimal eigenvalues, excluding the constant eigenvector.

HLLE performs well on those problems with partial non-convexity. One of the attracting features is its asymptotic stability. As the size of the input data increase, it will give better approximation of the embedding. A drawback of HLLE is that the Hessian approach requires estimation of second derivatives, and this is known to be numerically noisy or difficult in very high-dimensional data samples.

HLLE differs from the methods mentioned above in that it is seeking the recovery of the

underlying isometric coordinates of the manifold embedded in ambient space through the local coordinates. It is based on the observation that the quadratic form of the Hessian matrix on the manifold has $d + 1$ null space consisting a $d$-dimensional space of functions spanned by the original isometric coordinates. The linear construction of a given data point in LLE is replaced by a Frobenius norm of the Hessian matrix at that point. This local linear structure is copied to the low-dimensional space where the isometric coordinates reside based on the assumption of local isometry.

## 2.9 Local Tangent Space Alignment

Local Tangent Space Alignment (LTSA) [156] is a manifold learning algorithm benefitted from the assumption of local linearity. This leads to the result that the tangent space of each point on the manifold admits an orthonormal $d$-dimensional basis where its neighboring points will be projected onto and forms a set of local coordinates. To recover the underlying parameters which determine the manifold and hence the intrinsic low dimensional representation, these local coordinates are aligned by minimizing a set of local reconstruction error incurred by transforming the local coordinates to global ones

$$\mathbf{x}_{i_j} = \bar{\mathbf{x}}_i + \mathbf{L}_i \theta_j^i + \epsilon_j^i,$$

where $\mathbf{x}_{i_j}$ denotes the global coordinates of $j$th neighbor of $\mathbf{x}_i$, $\theta_j^i$ the corresponding local coordinates, $\mathbf{L}_i$ the local affine transformation matrix, $\epsilon_j^i$ the reconstruction error to be minimized and $\bar{\mathbf{x}}_i$ the mean of the $\mathbf{x}_i$ neighborhood. The neighborhood is determined by adjacency graph and the local coordinates are obtained by performing PCA on each neighborhood.

The overall reconstruction error can be reformulated in matrix form as

$$\sum_i \|\widetilde{\mathbf{X}}_i - \mathbf{L}_i \Theta_i\|_F^2$$

with $\widetilde{\mathbf{X}}_i$ the set of centered neighbors of $\mathbf{x}_i$ and $\Theta_i$ the set of local coordinates. Because of the quadratic form of the objective function and linearity assumption, the optimization with respect the $\mathbf{X}$ can be fulfilled by solving a eigensystem.

The idea of LTSA stems from the local coordinates alignment that is to find local coordinates based on some assumptions such as the local linearity in LTSA and then align them somehow in a global coordinate system. LTSA accomplishes it in deterministic manner by virtue of the manifold differentiation which is somewhat obscure. While some probabilistic treatments such as mixture of factor analysis [44, 39] is quite straightforward in coming up with the local coordinates. A series of DR methods follow this line of design [115, 137, 128, 17] which bring fresh air into the research of the DR. Several extensions of LTSA have been proposed later on such as supervised LTSA [85], linear LTSA [155], partitional LTSA [148], incremental LTSA [88] etc improving different aspects of the original LTSA.

## 2.10 Gaussian Processes Latent Variable Model

Gaussian Processes Latent Variable Model (GPLVM) [83] introduces the concept of Gaussian processes into unsupervised learning. It is derived from dual probabilistic PCA (DPPCA) through generalizing a linear model to any smooth function model. Each dimension of the input data is assumed to be governed by a Gaussian process parameterized only by a mean and a covariance function. Typically the mean is 0 and the covariance function is an RBF kernel. Maximizing the log-likelihood of $P(\mathbf{Y}|\mathbf{X})$ with respect to $\mathbf{X}$ gives us the optimal embeddings. Thus, it is equivalent to minimizing (suppose $\mathbf{Y}$ has been properly centered)

$$\frac{ND}{2}\ln 2\pi + \frac{D}{2}\ln|\mathbf{K}| + \frac{1}{2}\mathrm{Tr}(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^{\top}), \tag{2.18}$$

where $\mathbf{K}$ is the covariance matrix derived from the covariance function which is actually a kernel Gram matrix on embedded data. If we let $\mathbf{S} = D^{-1}\mathbf{Y}\mathbf{Y}^{\top}$, $\mathbf{S}$ can be seen as a linear kernel up to a scaling of $D^{-1}$, and hence can be interpreted as a Gaussian covariance on the input data. With two sets of measurements given by $\mathbf{S}$ and $\mathbf{K}$ in the two spaces, we can define two Gaussians $\mathcal{N}(0, \mathbf{S})$ and $\mathcal{N}(0, \mathbf{K})$. The Kullback-Leibler divergence between them can easily be calculated as follows

$$\mathrm{KL}(\mathcal{N}(0, \mathbf{S})||\mathcal{N}(0, \mathbf{K})) = \frac{1}{2}\ln|\mathbf{K}| - \frac{1}{2}\ln|\mathbf{S}| + \frac{1}{2}\mathrm{Tr}(\mathbf{S}\mathbf{K}^{-1}) - \frac{N}{2}. \tag{2.19}$$

It is recognized as a generalized objective function of (2.18) since the $\frac{1}{2}\ln|\mathbf{S}|$ and the other constant terms are independent of $\mathbf{X}$ and therefore contribute nothing in optimization. Thus, in GPLVM, the objective is to match the two Gaussians by minimizing the KL divergence between them. The optimization of (2.19) with respect to $\mathbf{X}$ ($\mathbf{X}$ is embedded in the expression of $\mathbf{K}$) is accomplished by gradient based methods such as conjugate gradient or scaled conjugate gradient algorithm [102].

As Gaussian process and probabilistic framework applied in GPLVM, several benefits come naturally. Because GPLVM actually defines a mapping from latent space to input space and GP provides smooth functions for that mapping, the extrapolation is feasible and easy. The other one is the ability to handle the missing value. As other probabilistic algorithms such as PPCA, GPLVM can process the lost attributes of the input data in a principle manner. Moreover, probabilities associated with the embeddings are also provided indicating the confidence that can be reposed in the results.

GPLVM actually enforces the dissimilarity matching, i.e., it focuses on keeping things apart in latent space that are far apart in data space. As such, GPLVM can be considered as dissimilarity preserving in some sense. But this property is not always desirable especially when input data lie on highly nonlinear manifold. [84] presented the GPLVM with back constraints where locality preserving is forced upon the algorithm and the out-of-sample extension is also achieved.

## 2.11 Kernel Principal Component Analysis

All the methods mentioned above assume that input data are in vectorial representation. Nevertheless, the applicability of DR algorithms to non-vectorial objects is demanded in advanced applications. Lacking efficient vectorised representations, the similarity among non-vectorial data cannot simply be evaluated by their Euclidean distances. By invoking the *kernel trick*, [120] made the first attempt to overcome the problem with non-vectorial input in a principled manner. The kernel trick reads: *given an algorithm which is formulated in terms of a positive definite kernel $\kappa$, one can construct an alternative algorithm by replacing $\kappa$ by another positive definite kernel $\tilde{\kappa}$.* Because dot product of normal vectors is actually the linear kernel, we simply replace

the dot product by a kernel. Since a positive definite kernel function admits a feature mapping by the fact that

$$\langle \phi(x), \phi(y) \rangle = \kappa(x, y),$$

where $x$ and $y$ are objects from general sets including non-vectorial data and $\phi()$ is the feature mapping, it can be viewed as having the kernelized algorithm working on the feature space owing to the mapping function.

Any linear methods involves only inner products could in theory be "kernalized" so that non-vectorial data can be properly processed. Kernel Principal Component Analysis (KPCA) [121] is a kernel version of PCA. It is a nonlinear generalization of PCA in the sense that PCA is performed in a feature space of arbitrarily large (possibly infinite) dimensionality via a kernel function which endows it the ability to handle non-vectorial data. Through the feature mapping associated with the kernel function, KPCA projects the mapped data onto the principal feature "vectors" in the feature space. Compared to standard PCA, KPCA solves a similar eigenvector problem as (2.6) with the inner product matrix $\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^\top$ being substituted by the centered kernel Gram matrix through applying the kernel trick:

$$N\lambda\boldsymbol{\alpha} = \widetilde{\mathbf{K}}\boldsymbol{\alpha}, \tag{2.20}$$

where $\widetilde{\mathbf{K}}$ is the centered kernel Gram matrix defined on the input data and $N$ the number of input objects. $\widetilde{\mathbf{K}}$ is centered by using the same technique in (2.8) where $\mathbf{A}$ is replaced by the original kernel Gram matrix $\mathbf{K}$, i.e. $\widetilde{\mathbf{K}} = \mathbf{HKH}$. The projections of the input data have the same form as (2.7) but $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_d$ are $d$ leading eigenvectors from solving (2.20). It is straightforward to get the coordinates for new input data $\mathbf{y}$,

$$\mathbf{x} = [\frac{\boldsymbol{\alpha}_1}{\sqrt{N\lambda_1}}, \ldots, \frac{\boldsymbol{\alpha}_d}{\sqrt{N\lambda_d}}]^\top \begin{bmatrix} \tilde{\kappa}(\mathbf{y}_1, \mathbf{y}) \\ \tilde{\kappa}(\mathbf{y}_2, \mathbf{y}) \\ \vdots \\ \tilde{\kappa}(\mathbf{y}_N, \mathbf{y}) \end{bmatrix}$$

where $\tilde{\kappa}(\cdot, \cdot)$ is the kernel function value after centering and $\lambda_i$'s are the eigenvalues associated with $\boldsymbol{\alpha}_i$'s.

As we have seen in Section 2.4 $\mathbf{X}$ is recovered from spectral decomposition of inner product of $\widetilde{\mathbf{Y}}\widetilde{\mathbf{Y}}^\top$ in classical MDS. If we replace the inner product matrix by $\widetilde{\mathbf{K}}$ and proceed the same procedure, we can have a set of coordinates. Through the analysis in section 2.4, we see that it is actually the result of KPCA whereas it is from the kernelized classical MDS.

As a union of kernel method and PCA, KPCA allows the simultaneous processing of non-vectorial data and the construction of their low-dimensional embeddings. The dimensionality of non-vectorial data is difficult to define as it is closely linked with feature selection and extraction. By resorting to the kernel trick, feature representation of the input data can be implicitly defined, relieving KPCA the necessity of any explicit features representation.

There are other linear DR methods suitable for kernelization. Linear Discriminant Analysis (LDA) [37] is a well accepted supervised linear DR though it was designed for classification. Its corresponding kernel version is the Generalized Discriminant Analysis (GDA) [41]. Nevertheless, kernel function itself is also a similarity or dissimilarity measure which applies to non-vectorial data. This feature can be exploited in DR. Some methods like Stochastic Proximity Embedding (SPE) [2] can accept kernel Gram matrices instead of vectors as input and produce the embeddings preserving the similarity relations expressed by the kernel. We generalize this idea to other DR methods such as LE. Rather than kernelizing it, we take kernal as a similarity measure and incorporate it in LE. As a result, we proposed the Kernel LE (KLE). We will discuss this method in detail in Chapter 4.

## 2.12 Summary

From the above analysis of several representative DR methods, we see that the implementation of DR varies by different point of view such as the global alignment of local coordinates (LTSA, HLLE etc). However, most of these methods are preserving some measures in both the input and latent space. Basically, a feature of the input data such as pairwise distances (MDS, Isomap etc), local neighborhood (LE, LLE etc), covariance (PCA, GPLVM etc) is extracted from the input data, and reproduced in corresponding embeddings. This procedure becomes the main stream in DR. The methods we proposed follow this design and process non-vectorial data as the

main goal. Before describing our methods, we will first present a kernel designed for shapes and images that will be frequently used in our methods in the next chapter.